Fundamentos de UNIX e Internet

Eric Raymond redactó en 1998 su clásico HOWTO introductorio, en el cual describe de manera concisa en un lenguaje no técnico las bases operativas de una computadora tipo PC, sistemas operativos tipo UNIX, y la Internet.

Tabla de contenido

1. Introducción

1.1. Propósito de este documento

1.2. Nuevas versiones de este documento

1.3. Comentarios y correcciones

1.4. Recursos relacionados

2. Anatomía básica de su computadora

3. ¿Que sucede cuando enciende la computadora?

4. ¿Que pasa cuando usted inicia sesión en el sistema?

5. ¿Que sucede cuando usted corre programas luego del arranque?

6. ¿Cómo funcionan los dispositivos de entrada y las interrupciones?

7. ¿Qué es lo que permite que mi computadora haga varias cosas a la vez?

8. ¿Que es lo que permite que mi computadora mantenga procesos de manera conjunta y organizada?

8.1. Memoria virtual: versión simple.

8.2. Memoria virtual: versión detallada.

8.3. La unidad de gestión de memoria.

9. ¿Qué es lo que permite que mi computadora guarde cosas en memoria?

9.1. Números

9.2. Caracteres

10. ¿Qué hace mi computadora cuando almacena cosas en el disco?

10.1. Nivel inferior del disco y el sistema de arhivos

10.2. Nombres de archivos y directorios

10.3. Puntos de montaje

10.4. ¿Cómo hace un archivo para ser visto?

10.5. Propietario, permisos y seguridad de archivo.

10.6. ¿Cómo las cosas pueden andar mal

11. ¿Cómo funcionan los lenguajes de programación?

11.1. Lenguajes Compilados

11.2. lenguajes Interpetados

11.3. Lenguajes Pseudocódigo

12. ¿Cómo funciona la Internet?

12.1. Nombres y Ubicaciones

12.2. El Sistema de Nombres de Dominio

12.3. Paquetes y enrutadores

12.4. TCP e IP

12.5. HTTP, un protocolo de aplicación

13. Para aprender más

inicio

---

1. Introducción

1.1. Propósito de este documento

Este documento tiene la intención de ayudar a los usuarios de Linux y de Internet quienes “aprenden haciendo” (learning by doing). Aunque esta es una gran manera de adquirir específicas habilidades, algunas veces nos deja problemas particulares en algunos conocimientos fundamentales. Estos problemas pueden dificultarnos la capacidad de pensar creativamente la solución efectiva de algunos problemas debido a la ausencia de un buen modelo mental sobre lo que realmente esta sucediendo.

Intentaré describir con un lenguaje simple y claro como trabaja todo esto. Este documento ha sido adecuado para personas que usan UNIX o Linux en hardware de computadoraas tipo PC. Aún así usualmente usaré el término 'UNIX' ya que la mayoría de lo que se describirá es consistente a través de las plataformas y las variantes de Linux.

Voy a asumir que usted esta usando un computadora tipo PC Intel. Los detalles difieren ligeramente si usted esta usando un Alpha, un PowerPC o algún otro UNIX, y que los detalles básico son los mismos.

No deseo repetir las mismas cosas, así que usted debe poner atención, pero esto no significa que debe aprender cada palabra que lea. Es buena idea solo hojear el documento cuando lo lea por primera vez, y regresar y releerlo unas pocas veces después de que haya digerido lo que haya leído.

Este es un documento en constante cambio. Intentaré crear nuevas secciones en la medida que los usuarios lo soliciten, así que regrese y revise periódicamente.

1.2. Nuevas versiones de este documento

Nuevas versiones de este documento (UNIX and Internet Fundamentals HOWTO / COMO de Fundamentos de UNIX e Internet) serán periódicamente publicadas en comp.os.Linux.help y comp.os.Linux.announce y news.answers. Estos documentos también serán subidos a varios sitios web y servidores FTP, incluyendo la página de inicio de LDP.

También puede visitar la última version de este documento en Internet en la dirección:https://tldp.org/HOWTO/UNIX-and-Internet-Fundamentals-HOWTO/index.html.

Este documento ha sido traducido al Farsi, Polaco y al Español y turco.

1.3. Comentarios, sugerencias y correcciones

Si usted tiene preguntas, comentarios o sugerencias sobre este documento,por favor sientase libre de escribir al email de Eric Raymond, en fulano@thyrsus.com. Acepto cualquier sugerencia o critica. Especialmente recibo links donde se encuentren detalladas explicaciones sobre conceptos individuales. SI usted encuentra un error en este documento, por favor hágamelo saber para corregirlo en la siguiente versión. Gracias.

Sobre esta traducción al español, comentarios, sugerencias y correcciones a peron@texto-plano.xyz.

1.4. Recursos relacionados

Si usted esta leyendo este documento buscando referencias sobre cómo hackear, usted también debe leer el Cómo Ser un Hacker, Preguntas y Respuestas / How To Become A Hacker FAQ, que contiene links hacia otros útiles recursos.

inicio

2. Anatomía básica de su computadora

Dentro de su computadora existe un chip de procesamiento que realiza realmente el proceso de cómputo. Este cuenta con una memoria interna (que la gente de DOS/Windows refiere como RAM" y la gente de UNIX a menudo la denomina "Núcleo"; este término para UNIX proviene de una vieja costumbre desde cuando las RAM estaban hechas con formas de donas con núcleo de ferrite). El procesador y la memoria se disponen en la Placa Madre (MotherBoard), que oficia de corazón de su computadora.

Su computadora cuenta de una pantalla y un teclado. Tiene un disco rígido y un CD-ROM y quizás unidades de discos flexibles. Algunos de esos dispositivos funcionan gracias a placas controladoras que se conectan en la placa madre y ayudan a la computadora a controlar estos dispositivos. Otros dispositivos funcionan directamente en la placa madre en forma de chipsets especializados, un conjunto de chips que cumplen función similar a la placas controladoras. El funcionamiento de su teclado es muy sencillo como para requerir una placa aparte; en este caso el controlador ya está incorporado directamente dentro del chasis del mismo teclado.

Más adelante nos adentraremos en algunos de los detalles de como trabajan estos dispositivos. Por ahora, veamos unas pocas cosas básicas para recordar sobre como estos dispositivos trabajan al unísono.

Todas las partes dentro del gabinete de su computadora están conectadas por un bus. Físicamente el bus es el lugar donde se insertan las placas controladoras (la placa controladora de video, la controladora de disco o si se tiene, la placa de sonido). El bus es la autopista entre su procesador, su pantalla, su disco y todo lo demás.

(Si usted ha visto referencias a 'ISA', 'PCI', y 'PCMCIA' relacionadas con las computadoras y no las ha entendido, estos son tipos de bus. ISA es -exceptuando algunas diferencias- el mismo bus que fue usado en los PC originales de IBM en 1980. Actualmente están en desuso. PCI (Periferical Component Interconnection / Componente de Interconexión Periférica) es el bus usado en la mayoría de los mas modernas computadoras así como en los más modernos computadores Machintosh. PCMCIA es una variedad de el bus ISA con conectores más pequeños, usado en computadoras portátiles.

El microprocesador, si bien hace que todo funcione, realmente no puede contactarse de forma directa con ninguno de los demás dispositivos, sino que debe comunicarse indirectamente con ellos recurriendo al bus. El único otro subsistema que es realmente rápido y con acceso inmediato al procesador es la memoria (el núcleo). Para que sus programas se ejecuten, deben estar **en** el núcleo (en la memoria)

Cuando su computadora lee un programa o datos del disco, lo que realmente sucede es que el microprocesador recurre al uso del bus para enviar a controlador de disco una solicitud de *lectura* de disco. Un instante después el controlador del disco recurre al bus para informar al microprocesador que ha leído los datos y los ha dispuesto en cierto lugar de la memoria. El microprocesador puede entonces usar el bus para enviar una solicitud de lectura al banco de memoria, y así mirar los datos

El teclado y la pantalla también usan el bus para comunicarse con el microprocesador, pero de una manera simple. La discutiremos más adelante, pero por ahora usted conoce lo suficiente como para entender qué sucede cuando enciende la computadora.

inicio

3. ¿Qué sucede al encender la computadora?

Una computadora sin un programa funcionando es solo un conjunto inerte de elementos electrónicos. La primer cosa que una computadora hace cuando se la enciende es iniciar un programa especial denominado Sistema Operativo. La labor del sistema operativo es ayudar a otros programas de la computadora a trabajar con los difíciles detalles de controlar el hardware de la computadora.

El proceso de encender el sistema operativo se llama Booting/arranque (originalmente el término fue bootstrapping/'calzarse las botas' en alusión al proceso de vestirse “comenzando desde los pies a la cabeza”). La computadora sabe cómo iniciar puesto que las instrucciones del arranque se encuentran definidas dentro de uno de sus chips, específicamente el chip BIOS (Basic Input/Output System, Sistema básico de entrada y salida)

El chip BIOS le indica a la computadora leer en un lugar específico, usualmente en el disco rígido numerado con el menor numero (disco de arranque) en busca de un programa especial llamado el 'boot loader'/'cargador del arranque' (en Linux, el cargador del arranque se llama Grub o LILO). EL cargador de arranque es copiado en el interior de la memoria e iniciado. El trabajo del cargador del inicio es es iniciar el verdadero sistema operativo.

El cargador de arranque lo logra haciendo lo mismo con el kernel, buscándolo, cargándolo en memoria e iniciándolo. Cuando usted inicia Linux y ve en pantalla el texto “LILO” (o “Grub”. n. del .t.) seguido de un grupo de puntos, significa que está cargando el kernel en memoria (cada punto significa que ha cargado otro bloque del disco del código del kernel.)

Podría preguntarse ¿porqué la BIOS no carga directamente el kernel desde un principio?, o bien ¿porqué el proceso del 'cargador del inicio' consta de estos dos pasos? La razón se debe a que la BIOS no es muy astuta... de hecho es bastante limitada, y Linux no vuelve a recurrir a la BIOS una vez transcurrido el momento del arranque. La BIOS fue escrita originalmente para primitivas computadoras de 8 bits con discos diminutos, y literalmente no puede acceder al disco lo suficiente como para cargar el kernel al completo. Además, el proceso de inicio también permite escoger iniciar uno de varios sistemas operativos que se encuentran ubicados en diferentes lugares del disco (en el caso poco probable que UNIX no le resulte lo suficientemente bueno).

Una vez el kernel arranca, debe otear los alrededores en busca del hardware restante, y alistarse para hacer funcionar los programas. El kernel no hace esto habitando en las direcciones ordinarias de memoria, sino que en vez de ello lo hace en los puertos de E/S ('Input/Output'/'Salida/Entrada)(E/S: estas son locaciones especiales en el Bus que probablemente están para que las placas de control de los dispositivos puedan escuchar las peticiones/ordenes del kernel. El kernel no hace esto de una manera desordenada; sino que dispone de una gran cantidad de información incorporada que detalla dónde y cómo responderán los controladores ante la posibilidad de estar presentes. Este proceso es llamado autoprobing / autodetección.

La mayoría de los mensajes que usted ve al momento del arranque son resultado de la autodetección de su hardware que realiza el kernel a través de los puertos de E/S, identificando cual esta disponible para cada dispositivo y adaptando dicho dispositivo a su máquina. El kernel Linux es extremadamente bueno en esto, mejor que la mayoría de los otros kernels UNIX y *mucho mejor* que el de DOS o Windows. En efecto, muchos veteranos de Linux consideran que la habilidad de Linux en la detección del proceso de arranque (lo cual hace que sea relativamente fácil para instalar) fue la principal razón para dejar de ser un paquete experimental y libre de UNIX y atraer a una masa critica de usuarios.

Pero dejar al kernel completamente cargado y funcionando no significa que ha terminado el proceso de arranque; esto consisten en sólo el primer paso del proceso (algunas veces llamado 'runlevel 1'/'nivel de funcionamiento 1'). Después del primer paso, un proceso especial llamado 'init' toma control del kernel, y despliega varios procesos de administración. (Algunos Linux recientes recurren a un programa diferente denominado 'upstart' que hace lo mismo).

La primer labor del proceso 'init' usualmente consiste en controlar para asegurarse que los discos estén en buen estado. El sistema de archivos del disco da cuenta de aspectos muy frágiles; si estuviese afectado por una falla del hardware o una interrupción súbita de alimentación eléctrica, serían buenas razones para tomar medidas de recuperación antes de que UNIX esté en completo

funcionamiento. Veremos algo de esto mas en detalle cuando hablemos sobre como puede fallar el sistema de archivos.

El siguiente paso del proceso 'init' es arrancar varios 'demonios'. Un demonio es un programa residente en memoria que funciona escondido en segundo plano, tal como la cola de impresión, el detector de llegada de emails o un servidor web, esperando cosas que hacer. Estos programas especiales a menudo deben coordinar varias peticiones que podrían entrar en conflicto. Estos procesos son 'demonios' por que a menudo es más sencillo escribir un programa que funcione constantemente y reconozca todas las peticiones que otro que intente asegurar que una multitud de copias de procesos (cada copia en proceso de una solicitud y todas funcionando al mismo tiempo) no interfieran unas con otras. El conjunto específico de demonios de arranque puede variar, pero casi siempre incluye un proceso de la cola de impresión (un demonio que oficia de patovica para su impresora).

El siguiente paso es prepararse para los usuarios. El proceso Init inicia una copia del programa **getty** dedicado a vigilar/atender su consola (y quizás mas copias para atender conexiones 'dial-in'/'por discado remoto' en puerto serial). En la actualidad, usualmente inicia múltiples copias de **getty** de manera de tener consolas virtuales (normalmente 7 u 9), con su pantalla y teclado conectadas en turno a una de ellas. Probablemente no vea ninguna de esas, si es que aún no ha alcanzado este segundo paso. Esto se debe a que una de sus consolas serán ocupadas por el servidor X (del cual contaremos en breve).

Pero no hemos terminado. El siguiente paso consiste en arrancar varios demonios que dan soporte a redes y otros servicios. El más importante de estos es su servidor X. X es un demonio que gestiona su pantalla, teclado y ratón. Su tarea principal es producir las gráficas de pixels a color que normalmente ve en su pantalla. Durante la última parte de su proceso de arranque de su máquina, cuando el servidor X se activa efectivamente, toma el hardware desde cualquier consola virtual que previamente estuviese en control. Este es el momento en el cual observará una pantalla gráfica de inicio de sesión, producida para usted por un programa denominado gestor de pantalla.

inicio

4. ¿Qué sucede al iniciar sesión en el sistema?

Cuando usted inicia sesión, se identifica a la computadora. En los UNIX modernos usualmente hará esto a través de un gestor de pantalla gráfico. Pero es posible cambiar a las consolas virtuales con una secuencia Ctrl+Nayúsculas y hacer un inicio de sesión de texto. En tal caso, usted lo hará a través de una instancia de **getty** observando a la consola llamar al programa **login**.

Frente al gestor de pantalla o a **login**, usted debe identificar un usuario con un nombre de usuario y una contraseña. Dicho nombre de usuario se somete a consulta en un fichero denominado /etc/passwd, que es una secuencia de líneas, cada una de ellas describe a una cuenta de usuario.

Uno de esos campos corresponde a una versión encriptada de la clave (algunas veces los campos encriptados son realmente almacenados en un segundo archivo /etc/shadow con permisos mas astringentes haciendo que sea difícil descubrir la clave). Lo que usted mecanografía como clave resulta encriptada exactamente de la misma forma y el programa **login** compara si son iguales. La seguridad de este método esta en el hecho que, mientras que es fácil ir de la versión limpia de la

clave a la versión encriptada, el método inversa es muy difícil. De esta manera, aún si alguien puede ver la versión encriptada de la clave no le será posible usarla. (esto también significa que si usted olvida su clave no hay forma de recuperarla, solo funcionará cambiarla a algo que usted escoja.)

Una vez que ha iniciado sesión satisfactoriamente, usted obtiene todos los privilegios asociados con la cuenta específica que usted esté usando. También puede ser reconocido como parte de un 'grupo'. Un grupo es una colección de nombres de usuarios establecida por el administrador del sistema. Los grupos pueden tener privilegios independientes de los privilegios de sus miembros. Un usuario puede ser miembro de múltiples grupos. (para detalles sobre como trabajan los privilegios en UNIX vaya a la sección de permisos.

Note que aunque usted se referirá normalmente a usuarios y grupos por un nombre, realmente ellos internamente se encuentran almacenados internamente en forma de identificadores numéricos (IDs). El archivo de clave compara su nombre de cuenta contra un identificador de usuario; el archivo /etc/group compara los nombres de grupo contra un identificador de grupo. Los comandos que negocian con las cuentas y los grupos realizan automáticamente la traducción.

Su inicio de cuenta también contiene un 'directorio de inicio', el lugar donde el sistema de archivos de UNIX ubicará sus archivos personales. Finalmente su cuenta de usuario también ajustará su shell, el interprete de comandos que el programa **login** arrancará para aceptar sus comandos.

Lo que sucede luego que ha iniciado sesión exitosamente depende de cómo lo hizo. En una consola de texto, el programa emulador de terminal que corra una shell.

inicio

5. ¿Qué sucede al correr programas luego del arranque?

Desde el momento en que el sistema arranca, y una vez que usted corre un programa, puede pensar en su computadora como una jaula de un zoológico de procesos que están esperando algo que hacer. Ellos están en la espera de eventos. Un evento puede consistir en la presión de una tecla o el movimiento del ratón. O si su máquina está enlazada a una red de datos, un evento podría ser un paquete de datos entrando en la red.

El kernel es uno de dichos procesos. Se revela especial pues controla cuando pueden correr otros procesos de usuario, y es normal que sea el único proceso que cuenta con acceso directo al hardware de la máquina. En efecto, cuando los demás procesos de usuario desean conseguir una entrada de teclado, escribir en su pantalla, leer el disco o escribir en él, o solamente hacer cualquier otra cosa con los bits en memoria, deben realizar una petición al kernel. Estas peticiones se conocen como llamadas de sistema.

Normalmente todas las entradas y salidas (E/S) pasan a través del kernel así que el puede programar las operaciones y prevenir que los procesos se pisen entre sí. Unos pocos procesos de usuario tienen permitido circunvalar al kernel, usualmente por que se les ha dado acceso directo a los puertos de E/S. Los servidores X constituyen el ejemplo más común de este proceder.

Usted correrá programas en una de dos formas posibles: a través de su servidor gráfico X o a través de la shell del intérprete de comandos. A menudo, podrá hacer ambas cosas, puesto que iniciar un emulador de terminal imita una consola de texto clásica, dándole una shell para correr programas.

Describiré qué sucede cuando hace esto, y luego volveremos a lo que sucede cuando corre un programa a través de un menú de X o un ícono de escritorio.

El intérprete de comandos se denomina *shell* porque envuelve y oculta el kernel del sistema operativo. Es una característica importante de UNIX que la shell y el kernel sean programas separados que se comunican a través de un conjunto de solicitudes de llamadas de sistema. Esto hace posible que existan múltiples shells, que sirvan a distintos gustos de interfaz.

La shell normal le presenta el prompt '$' que ve luego de iniciar sesión (a no ser que haya personalizado esto para que presente otra cosa). No nos referiremos a la sintaxis de shell y las cosas fácil que puede hacer en la pantalla aquí; en su lugar hecharemos un vistazo en la trastienda de lo que sucede desde el punto de vista de su computadora.

La Shell es solo un proceso de usuario, sin nada en particular. Aguarda la pulsación del teclado, oyendo (a través del kernel) al puerto E/S del teclado. En la medida que el kernel los oye, los repite como eco presentándolo en su consola virtual o en su emulador de terminal de X. Cuando el kernel recibe la pulsación de la tecla 'Intro', introduce su linea de texto a la shell. La shell intenta interpretar los caracteres tecleados como si fuesen comandos.

Digamos que usted teclea 'ls' y pulsa la tecla 'Intro' para invocar el listador de directorios de UNIX. La shell aplica sus reglas preestablecidas para entender que usted desea correr el comando u orden de ejecución en el archivo */bin/ls*. El comando hace una llamada de sistema solicitando al kernel iniciar */bin/ls* como un nuevo proceso hijo, dándole acceso a la pantalla y el teclado a través del kernel. Tras ello, la shell dormitará esperando que ls finalice.

Cuando la orden /bin/ls finaliza, avisa al kernel su finalización emitiendo al sistema una petición de sistema *exit*. El kernel entonces despierta a la shell diciéndole que puede continuar con su propia ejecución. La shell emite otro aviso y espera por otra entrada.

Sin embargo otras cosas pueden estar sucediendo mientras se ejecuta su orden 'ls' (supongamos que usted esta listando un directorio muy largo). Usted podría querer cambiar a otra consola virtual, loguearse en ella, e iniciar - por ejemplo - un juego de Quake. O suponiendo que usted esta enlazado a la Internet, su máquina podría encontrarse enviando o recibiendo correo electrónico mientras de ejecuta /bin/ls.

Cuando usted corre programas a través del servidor X en lugar de hacerlo desde la shell (o sea, seleccionando una aplicación desde un menú pull-down, o haciendo doble clic sobre un ícono de escritorio), cualquiera de los varios programas asociados con su servidor X puede comportarse de la misma forma que una shell, y lanzar el programa. Voy a descglosar los detalles aquí porque son variables y poco importantes. El punto es que el servidor X, a diferencia de una shell normal, no es puesto a dormir mientras que programa cliente se encuentra en ejecución. En lugar de ello, queda dispuesto entre usted y el cliente, sopesando sus clics de ratón y presiones de teclas y cumplimentando las solicitudes de pintar pixels en su pantalla.

inicio

6. ¿Cómo funcionan los dispositivos de entrada y las interrupciones?

Su teclado es un dispositivo de entrada muy simple; simple porque genera pequeñas cantidades de datos muy lentamente (por un estándar de computación). Cuando usted presiona o libera una tecla, se envía un evento a travpes del cable del teclado al puerto del mismo para provocar una 'interrupción de hardware' / 'hardware interrupt'.

El trabajo del sistema operativo es atender este tipo de interrupciones. Para cada tipo de interrupción posible existe un 'manipulador de interrupción' / ' interrupt handler', como parte del sistema operativo que acoge cualquier dato asociado con dicho manipulador de interrupción (como su valor de presión de tecla/soltado de tecla) hasta que pueda ser procesado.

Lo que el manipulador de interrupciones hace realmente con su teclado es enviar el valor de tecla a un área de sistema cercano al inicio de la memoria. Allí, el valor estará disponible para cuando el sistema operativo pase el control a cualquier programa que se supone esta actualmente leyendo el teclado.

Dispositivos de entrada mas complejos como los discos o la placas de red operan de manera similar. Anteriormente hice referencia a un controlador de disco que usaba el bus para señalar la realización de una petición al disco. Lo que realmente sucede es que el disco recibe una interrupción. El manipulador de interrupciones del disco copia entonces los datos recibidos a la memoria para uso posterior por parte del programa peticionante.

Cada tipo de interrupción tiene asociado un 'nivel de prioridad' / 'priority level'. Interrupciones con *prioridades bajas* (como los eventos del teclado) tienen que esperar por/sobre las interrupciones con *prioridades altas* (como los tics del reloj o los eventos del disco. Como se necesita mantener aceleradas las respuestas que da la máquina, UNIX esta diseñado para dar prioridad al tipo de eventos que necesitan ser procesados rápidamente.

En los mensajes que aparecen en el momento del arranque, usted puede ver referencias a números IRQ. Podría ya estar al tanto que una de las típicas formas de hacer una incorrecta configuración de hardware es teniendo a dos dispositivos distintos intentando usar el mismo IRQ, sin entender exactamente por que.

He aquí la respuesta. IRQ es la abreviatura de “Petición de Interrupción” / “Interrupt Request”. El sistema operativo necesita saber en el momento del arranque qué número de interrupción usara cada dispositivo de hardware, para asociar adecuadamente los manipuladores de interrupción. Si dos dispositivos distintos tratan de usar una misma IRQ, las interrupciones serían enviadas al manipulador equivocado. Como mínimo esto inhabilita el dispositivo y la mayor de las veces puede confundir negativamente al sistema, lo suficiente como para saturarlo o anularlo.

inicio

7. ¿Qué es lo que permite que mi computadora haga varias cosas a la vez?

Realmente no lo hace. Las computadoras solamente pueden obrar una *tarea* (o proceso) a la vez. Sin embargo, la computadora puede conmutar entre tareas muy rápidamente, y engañar a los lentos seres humanos llevándolos a pensar que está haciendos varias cosas a la vez. A esto se le llama tiempo de cómputo compartido / 'timesharing'

Una de las tareas del kernel es administrar la compartición de tiempo de cómputo. El tiene una parte llamada el programador / scheduler, el cual retiene información sobre los otros procesos (no de kernel) de su zoológico. Cada 1/60 de segundo un temporizador se dispara en el kernel, generando una interrupción de reloj. El programador detiene cualquier proceso en ejecución, actual, suspende su ejecución, y cede control a otro proceso.

1/60 de segundo puede no sonar a mucho tiempo. Pero para los microprocesadores actuales este tiempo es suficiente para correr decenas de miles de instrucciones de máquina, las cuales pueden significar mucho trabajo. Incluso si usted dispone de muchos procesos, cada uno de ellos puede completar bastantes cosas en sus intervalos de tiempo asignados.

En la practica, puede que un programa no logre usar la totalidad de su intervalo de tiempo asignado. Si una interrupción llega desde un dispositivo E/S, el kernel detendrá la tarea actual, iniciará el manipulador de interrupción y retornará a la actual tarea. Una tormenta de interrupciones de alta prioridad bien podría exprimir el normal procesamiento: esta mala conducta es llamada 'thrashing' y por fortuna es muy difícil de inducir en los modernos UNIX

En efecto, la velocidad de los programas es solo muy raramente limitada por la cantidad de tiempo de máquina que pueden obtener (existen algunas excepciones a esta regla, como las generación de sonido y gráficos 3D). Las demoras suceden mas a menudo cuando un programa debe aguardar el arribo de un dato desde una unidad de disco u una conexión de red.

Un sistema operativo que rutinariamente es capaz de correr muchos procesos simultáneamente es llamado “multitarea” / “multitasking”. La familia de sistemas Operativos UNIX fueron diseñados desde sus cimientos para la multitarea y UNIX en particular es muy bueno en ello – mucho mas efectivo que Windows o el viejo MacOS, los cuales le atornillaron la multitarea a último momento y la efectuaban de manera muy pobre. Su acción de multitarea eficiente y confiable es en gran parte lo que hace a Linux superior para las redes de datos, comunicaciones y servicios web.

inicio

8. Qué es lo que permite que mi computadora mantenga procesos de manera conjunta y organizada?

El programador del kernel cuida de dividir los procesos en el tiempo. Su sistema operativo también tiene que dividir estos procesos en el espacio, de tal forma que los procesos no puedan montarse unos con otros. Aun si usted supone que todos los programas intentan trabajar de manera cooperativa, no desearía que un bug (error) en uno de ellos se viese capaz de corromper a otros. Las

operaciones que su sistema operativo realiza para resolver este tipo de problemas se denominan 'administración de memoria' / 'memory management'.

Cada proceso en su máquina necesita su propia área de memoria, como un lugar donde correr su código y mantener sus variables y resultados. Usted puede pensar que este conjunto de cosas esta compuesta de un segmento de código de solo lectura (que contiene las instrucciones de los procesos) y un segmento de datos escribible (que contiene todas las variables del proceso). El segmento de datos resulta único para cada proceso, pero si dos procesos corren el mismo código al unísono, UNIX automáticamente los organiza para que compartan el mismo segmento de código, como una medida de eficiencia.

8.1. Memoria virtual: versión simple

La eficiencia es importante por que la memoria es costosa. *En ciertos casos* podría contar con memoria insuficiente para dar soporte a todos aquellos programas que está corriendo la máquina, especialmente si usted emplea programas de gran tamaño, tales como un servidor X. Para dar cuenta de estos casos UNIX recurre a una técnica llamada Memoria Virtual. En lugar de intentar almacenar todo el código y datos necesarios para un proceso, UNIX mantiene en memoria sólo un espacio de trabajo relativamente pequeño; el resto del proceso es dejado en un área especial en su disco rígido denominada 'espacio de intercambio'/'swap space'.

Tenga presente que en el pasado el término "*en ciertos casos*" significaba "*casi siempre*", como consecuencia de que el tamaño de la memoria era generalmente pequeño en relación al tamaño de los programas en ejecución, de modo que era sumamente frecuente recurrir al espacio de intercambio ("swapping"). Hoy en día la memoria es mucho menos costosa y aun aquellas máquinas de bajas prestaciones cuentan con menoria suficente. Un computador personal de 64 MB de memoria o mas le posibilita correr servidores X y un conjunto de tareas típico sin tener que recurrir casi nunca al espacio de intercambio (una vez que se han cargado en memoria inicialmente).

8.2. Memoria virtual. Versión detallada

Realmente en la última sección se simplificaron las cosas un poco. Sí, los programas ven la mayor parte de la mayoría como un banco de direcciones de memoria unívoco más grande que una memoria física, y se usa un disco como 'memoria de intercambio' para mantener esta ilusión. Pero en todo caso su hardware tiene no menos que cinco diferentes tipos de memoria distintos dentro de si, y la diferencia entre estos tipos de memoria puede ser importante cuando los programas deben ser afinados para que logruen la máxima velocidad posible. Para entender realmente que sucede dentro de su computadora, usted debe aprender como todas ellas trabajan en conjunto.

Los cinco tipos de memoria son estos: registros del procesador, caché interno (o en chip), caché externo (o fuera de chip), memoria principal, y disco. Y la razón de que sean varios tipos es simple: la velocidad cuesta dinero. He listado estos tipos de memoria en orden creciente respecto a el tiempo de acceso y en orden decreciente respecto al costo. La memoria de registros del procesador es la mas rápida y costosa, y puede ser accedida de manera aleatoria en centenares de millones de veces por segundo, mientras que la memoria del disco es la mas lenta y barata, y puede realizar cerca de 100 accesos aleatorios en un segundo.

He aquí una lista que refleja velocidades datadas en el inicio de los años 2000 de una máquina de escritorio común. Mientras que la velocidad y la capacidad crecen los precios van bajando y es posible esperar que este tipo de proporciones se sostengan de manera constante.

Disco: Tamaño 13000 MB. Velocidad de acceso: 100KB/seg.

Memoria principal: Tamaño 256 MB. Velocidad de acceso: 100MB/seg.

Cache Externo: Tamaño 512 MB. Velocidad de acceso: 250MB/seg.

Cache Interno: Tamaño 32KB. Velocidad de acceso: 500MB/seg.

Procesador: Tamaño 28 bytes. Velocidad de acceso: 1000MB/seg.

No es posible construir todo a partir de los más rápidos tipos de memoria. Se volvería demasiado caro - y aun si no lo fuera - la memoria rápida es volátil. Esto significa que pierde su sustento cuando la energía desaparece. Es por ello que las computadoras deben contar con discos rígidos u otros tipos de memorias no volátiles que retengan los datos almacenados cuando se desconecta la alimentación. Y existe además un gran desequilibrio entre la velocidad del procesador y la velocidad del disco. Los tres niveles de en medio de la jerarquía de memorias (cache interno, cache externo y memoria principal) existen básicamente para cerrar esta brecha.

Linux y otros UNIX tienen una característica llamada 'memoria virtual' / 'virtual memory'. Lo que esto significa es que el sistema operativo se comporta como si tuviera mas memoria de la que realmente tiene. Su verdadera memoria física principal se comporta mas como un gran conjunto de ventanas o cajones (caches) que un gran espacio de memoria “virtual”, la mayor parte de la cual en un momento dado se encuentra almacena en disco en una zona especial llamada 'swap area' / 'zona de intercambio'. Fuera del alcance de los programas de usuario, el sistema operativo se encuentra moviendo bloques de datos (conocidos como “páginas”) entre la memoria y el disco para mantener esta ilusión. El resultado final es que su memoria virtual es mucho mas grande pero no mas lenta que la memoria real.

Qué tan lenta sea la memoria virtual con respecto a la memoria física depende de la manera en que los algoritmos de intercambio del sistema operativo den respuesta a la forma en que los programas usan la memoria virtual. Afortunadamente los datos leídos y escritos por la memoria que están cerca en tiempo también tienden a estar cerca en el espacio de memoria. Esta tendencia es llamada 'Localidad' / ' Locality' o mas familiarmente 'localización de referencia' (y esto es algo bueno). Si las referencias de memoria saltaran alrededor de la memoria virtual de manera aleatoria, generalmente se tendría que leer y escribir del/en disco para cada nueva referencia, y la memoria virtual sería tan lenta como lo es el disco. Pero como los programas suelen manifestar una localidad arraigada, el sistema operativo usualmente requiere realizar relativamente escasos swaps/intercambios por referencia.

Por experiencia se descubrió que el método más efectivo para afrontar un amplio tipo de patrones de uso de memoria es muy simple: el llamado LRU o el algoritmo del "último uso reciente". El sistema de memoria virtual toma los bloques del disco dentro de su *conjunto de trabajo* en la medida que lo necesite. Cuando agota su memoria física para satisfacer al espacio de trabajo,

descarga el bloque 'mas recientemente usado'. Todos los UNIX y la mayoría de los otros sistemas operativos con memoria virtual usan pequeñas variaciones del LRU.

La memoria virtual es el primer vínculo en el puente entre las velocidades del disco y del procesador. Esta explícitamente administrada por el sistema operativo. Pero aun existe un gran diferencial entre la velocidad de la memoria principal y la velocidad en la cual el procesador puede acceder a su memoria de registro. Las memorias cache internas y externas apuntan a remediar esto utilizando una técnica similar a la que hemos descrito sobre la memoria virtual.

Así como la memoria principal se comporta como un conjunto de ventanas y cajones en el área de intercambio/swap del disco, el caché externo actúa como ventanas sobre la memoria principal. El caché externo es rápido (250M de velocidad de acceso por segundo, en lugar de 100M) y pequeño, El hardware (específicamente el controlador de la memoria de su computadora) realiza el LRU en el caché externo sobre los bloques de datos traídos desde la memoria principal. Por razones históricas la unidad de almacenamiento de caché de intercambio se conoce como 'linea' en lugar de 'página'.

Pero aún no terminamos. El caché interno nos da el último incremento de velocidad efectiva al almacenar porciones de la caché externa. Es mas rápido y pequeña aún; de hecho esto ssucede en el chip del procesador.

Si realmente queremos hacer que nuestros programas sean realmente rápidos, nos será muy útil conocer estos detalles. Sus programas lograrán rapidez cuando tienen una fuerte localidad, por que esto hace que el caching trabaje mejor. La manera MÁS FÁCIL de hacer que los programas sean rápidos es, por lo tanto, hacerlos mas pequeños. Si un programa no resulta enlentecido por emplear grandes cantidades de E/S de disco o espera de eventos de red, usualmente correrá a la velocidad del cache mas pequeño en el que pueda caber.

Si usted no puede hacer que la totalidad de su programa sea pequeño, enfoque los esfuerzos en ajustar la localidad de las porciones con problemas de velocidad. Los detalles sobre las técnicas para hacer esto van mas allá del alcance de este tutorial: En el momento que usted las necesite, usted habrá profundizado lo suficiente con algún compilador para entender por si mismo muchas de las técnicas.

8.3. La unidad de administración de memoria

Aun cuando usted tenga suficiente memoria principal como para evitar el swapping, la parte del sistema operativo llamada 'administrador de memoria'/'memory manager' tiene aun importantes cosas que hacer. Debe asegurarse que los programas puedan solamente alterar sus propios segmentos de datos, esto es prevenir que códigos erróneos o maliciosos dentro de un programa dañen los datos de otro. Para llevarlo a cabo, él registra una tabla de datos y de segmentos de código. Esta tabla resulta actualizada siempre que un proceso cualquiera realice una petición de mas memoria o libere memoria (usualmente cuando el programa termina).

Esta tabla es usada para pasar ordenes a una especializada parte esencial del hardware llamada MMU o 'unidad de administración de memoria'. Los chips de procesadores modernos tienen MMUs integrados directamente en ellos. La MMU tiene la especial habilidad de colocar barreras alrededor de las áreas de la memoria, de tal forma que una referencia fuera de limite será rechazada y causará una situación de interrupción especial.

Si alguna vez observa un mensaje de UNIX que dice “Segmentation fault”/”falla de segmentación”, “core dumped”/”volcado de núcleo”, o algo similar, esto es lo que exactamente esta sucediendo: un programa en ejecución ha hecho una solicitud para acceder a la memoria (núcleo) fuera de su segmento dando así lugar a una interrupción fatal. Esto indica un error en el código del programa: el '*volcado de núcleo*' deja tras de si una información de diagnostico destinada a ayudar al programador a detectar el error.

Existe otro aspecto para proteger procesos de los otros además de limitar sus accesos de memoria. Usted también querrá ser capaz de controlar sus accesos a archivos para que un programa malicioso o problemático no corrompa piezas criticas del sistema. Esta es la razón del por que UNIX tiene permisos de archivos, tema que discutiremos mas adelante.

inicio

9. Que es lo que permite que mi computadora guarde cosas en memoria?

Usted probablemente sabe que cualquier cosa en un computadora es almacenada como una cadena de bits (dígitos binarios: usted puede pensar en ellos como un gran número de pequeños interruptores de encendido y apagado). Aquí vamos a explicar como estos bits son usados para representar las letras y números que su computadora mastica.

Antes de comenzar con esto, se necesario entender el “tamaño de palabra”/”word size” de su computadora. El tamaño de palabra es tamaño preferido de su computadora para manipular unidades de información: Técnicamente es el ancho del registro de su procesador, área usada por el procesador para hacer cálculos lógicos y matemáticos. Cuando la gente escribe sobre computadoras con distintos tamaños en bits (llamándolas computadoras de “32 bits” o “64 bits”), esto es lo que significa.

La mayoría de las computadoras tiene un tamaño de palabra de 64 bits. En el pasado reciente (comienzos de los años 2000s) muchas PCs tenían un tamaño de palabra de 32 bits (incluyendo computadoras 386, 486 y Pentium). Los viejas máquinas 286 de los 80s tenían un tamaño de palabra de 16: Las viejas mainframes a menudo usaban palabras de 36-bit.

La computadora ve su memoria como una secuencia de palabras numerada desde cero hasta algún alto valor dependiendo del tamaño de su memoria. Ese valor esta limitado por el tamaño de palabra. Esa es la razón por la cual los programas en viejas máquinas como las 286 tenían que funcionar mediante difíciles piruetas para dirigir grandes cantidades de memoria. No las describiré aquí; aún dan pesadillas a los viejos programadores.

9.1. Números

Los números enteros son representados como cualquier otra palabra o pares de palabras, dependiendo de el tamaño de palabra de su procesador. Una máquina con 64 bits es la mas común representación de entero.

La aritmética del entero es cercana pero no es realmente matemática base 2. Los bit de bajo orden son 1, le sigue el 2, el 4 y así sucesivamente como un puro binario. Pero los números con signo son

representados en la notación de “complemento a dos”. Los bits de mas alto orden son un bit con signo, lo cual hace que la cantidad sea negativa y cada numero negativo puede obtener su correspondiente positivo valor al invertir todos los bits y sumando uno. Es por esto que los enteros en una máquina de 64 bits tienen el rango de -2^64 a 2^64 – 1. Ese 64avo bit esta siendo usado para el signo: 0 significa un número positivo y 1 significa un numero negativo.

Algunos lenguajes de computación permiten usar la aritmética sin signo, lo que significa base 2 con cero y números positivos solamente.

La mayoría de los procesadores y algunos lenguajes pueden hacer operaciones con 'números de coma flotante'/'Flotating-point number' (esta capacidad esta preestablecida dentro de los chips de procesadores mas recientes). Los números de coma flotante le dan a usted rangos de valores más amplios que los enteros y permiten ser expresados en fracciones. La forma en la cual se realiza esto varía, y sería muy complicada de discutir ahora en detalle, pero en ideas generales se refiere a la 'notación científica', donde uno podría escribir 1.234 * 1023; la codificación de el numero es dividida en una *mantisa* (parte decimal de la fracción. n del t.) y la parte del exponente (23) para la potencia decimal (lo que significa que el número multiplicado resultante podría tener 20 ceros. 23 menos los tres lugares decimales.

9.2. Caracteres

Los caracteres son normalmente representados como cadenas de siete bits cada carácter en una codificación llamada ASCII (American Standard Code for Information Interchange / Estandard Americano de Código para el Intercambio de Información): en las máquinas modernas, cada uno de los 128 caracteres ASCII son los últimos (los mas bajos) siete bits de un octeto o byte de 8 bits: los octetos son empaquetados dentro de las palabras de memoria, así que (por ejemplo) una cadena de seis caracteres sólo puede tomar dos palabras de memoria. Para ver un listado con los códigos ASCII escriba 'man ascii' en su ventana de UNIX.

El anterior párrafo puede causar confunsiones en dos formas. La menor de ellas es que el término 'octeto' es formalmente correcto pero en realidad raramente es poco usado: la mayoría de las personas se refieren a un octeto como un byte y espera que tengan 8 bits de longitud. Estrictamente hablando, el término 'byte' es más general: es usado para, por ejemplo máquinas de 36 bits con bytes de 9 bits (aunque esto probablemente nunca sucederá de nuevo)

La mayor de ellas es que no está permitido que todo el mundo usa ASCII. En efecto, muchas de las palabras que existe no pueden ser escritas en ASCII. Esto puede no ser importante par el Inglés Americano, pero los acentos y también otros caracteres especiales son necesarios para los usuarios que usan otros idiomas. Inclusive el Inglés Británico tiene problemas con la ausencia del símbolo usado para el símbolo de la moneda Libra Esterlina.

Han existido varios intentos de corregir este problema. Todo uso de bits de mayor tamaño que no esta en ASCII, haciéndolo en la mitad inferior del conjunto de 256 caracteres. El mayor uso de esta técnica es llamado 'conjunto de caracteres Latin-1 (mas familiarmente llamado ISO 8859-1). Este es el conjunto de caracteres por defecto para Linux, HTML y X. Microsoft Windows usa una versión alterada de Latin-1 adicionándole caracteres como las dobles comillas derecha e izquierda en lugares donde el propio Latin-1 ha dejado sin asignar por históricas razones (para profundizar sobre los problemas que esto causa visite la pagina demoroniser)

Latin-1 permite usar idiomas de Europa Occidental, incluyendo el Inglés, francés, Alemán, Español, Italiano, Alemán, Noruego, Suizo, Danés e Islandés. Sin embargo esto no es suficiente, y como resultado ahora hay toda una serie de comjuntos de caracteres que van desde el latin-2 hasta el 9 para dar soporte a los caracteres del griego, árabe, Hebreo, Esperanto y serbo-croacia. Para mayores detalles visite la página Sopa del alfabeto ISO .

La última solución es un gran estandard llamado Unicode (y su gemelo idéntico ISO/IEC 10646-1:1993). Unicode es idéntico al Latin-1 en los 256 casilla mas bajos. Sobre estos hay un espacio de 16 bits que incluye el Griego, Cirilico, Armenio, Hebreo, Arabe, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Thai, Lao, Georgiano, Tibetano, Kana japonés, el conjunto de caracteres completo del Hangul Coreano y el conjunto de caracteres unificado para los ideogramas Chinos/Japoneses/Koreanos (CJK). Para mayores detalles visite la página de Unicode. XML y XHTML usan este conjunto de caracteres.

Las versiones recientes de Linux usan un codificado de Unicode llamado UTF-8. En UTF, los caracteres 0-127 son ASCII. Los caracteres 128-255 son usados sólo en secuencias de 2 a 4 bits que identifican caracteres no ASCII.

inicio

10. ¿Qué hace mi computadora cuando almacena cosas en el disco?

Cuando observa un disco rígido bajo UNIX, usted apreciará la existencia de un árbol de directorios y archivos con nombre. Normalmente usted no necesita profundizar en su estructura, pero él se ha convertido en algo muy útil para saber que sucede en detalle si usted tiene un disco dañado y necesita recuperar los archivos. Desafortunadamente, no es una buena forma describir la organización del disco desde el nivel inferior del mismo, así que lo describiremos desde el mismo hardware hacia arriba.

10.1. Disco de bajo nivel y Estructura de sistema de archivos

El área de superficie de su disco, donde él almacena la información, esta dividida en algo similar a un tablero de dardos – con pistas concéntricas divididas en sectores similares a las porciones de un pastel. Debido a que las pistas cercanas a la parte exterior tienen un área mayor que aquellas cercanas al centro del disco, las pistas del exterior se encuentran divididas en mas sectores que las pistas del interior. Cada sector (o bloque de disco) tiene el mismo tamaño, el cual bajo los modernos UNIX es generalmente 1 KB binario (1024 bytes de 8 bits). Cada bloque de disco tiene su única dirección o 'disk block number'/'numero de bloque del disco'.

UNIX divide el disco en 'particiones de disco'. Cada partición es un campo continuo de bloques que son usados separadamente de cualquier otra partición, cada uno como un sistema de archivos o espacio de intercambio aparte. La razón original para tener particiones es la de recuperar los daños en un mundo de discos muy lentos y susceptibles a errores; los limites entre estas particiones reducen la fracción de su disco que probablemente lo haría inaccesible o corrupto debido a un mala lectura del disco. Hoy en día es más importante que las particiones puedan ser declaradas como 'solo-lectura' (previniendo a un intruso el modificar archivos críticos del sistema) o compartirlas

sobre una red de varias maneras que no discutiremos aquí. La partición con el número más bajo es a menudo tratada de manera especial como la partición de arranque'/'boot partition' donde usted puede colocar el kernel para que sea iniciado.

Cada partición es implementada como espacio de intercambio (usado para implementar la memoria virtual) o un sistema de archivos para almacenar archivos. Las particiones de intercambio corresponden a una secuencia lineal de bloques. Por otro lado, el sistema de archivos necesita una forma de mapear(relacionar) los nombres de los archivos a secuencias de bloques del disco. Debido a que los archivos crecen o disminuyen de tamaño y/o cambian con el tiempo, un bloque de datos de archivos no sería una secuencia lineal de estos sino mas bien todpodrían encontrarse de manera dispersa a lo largo de su partición. (desde donde quiera que el sistema operativo pueda encontrar un bloque libre cuando lo necesite). Este efecto de dispersión se llama 'fragmentación'.

10.2. Nombres de archivos y directorios

Dentro de cada sistema de archivos, el proceso de mapeo de nombres a bloques es manipulado a través de una estructura llamada '*i-node*'. Hay un conjunto de ellos cerca de la parte inferior (los números más bajos de bloque) de cada sistema de archivos (los más bajos de ellos son usados para proteger y etiquetar propósitos que no describiremos aquí). Cada I-node describe un solo archivo. Los bloques de datos de archivos (incluyendo directorios) se encuentran sobre los i-nodes (los números más altos del bloque)

Cada I-node contiene una lista de números de bloque del disco en el archivo que describe (Actualmente esto no es del todo cierto, solo es correcto para archivos pequeños, pero el resto de los detalles no es importante conocerlos aquí). Note que cada i-node no contiene el nombre del archivo.

Los nombres de los archivos se encuentran en la 'estructura de directorios'. Una 'estructura de directorios' solo mapea los nombres hacia los números i-node. Esto sucede porque, en UNIX, un archivo puede tener múltiples nombres y todos verdaderos (a esto se le llama Links Duros): Estos son solo múltiples entradas de directorios que apuntan al mismo i-node.

10.3. Puntos de montaje

En la situación más simple, su sistema de archivos de UNIX entero podría encontrarse en una sola partición. Esto es inusual aun cuando usted vea esto en algunos pequeños sistemas personales UNIX. Es más típico que estén regados a lo largo de varias particiones, posiblemente en diferentes discos físicos. Así que, por ejemplo, su sistema puede tener una pequeña partición donde se encuentra el kernel, una partición ligeramente mayor donde se encuentran las utilidades del sistema operativo, y una partición mucho más grande donde se encuentran ubicados los archivos del usuario.

La única partición a la cual se tiene acceso inmediatamente después de arrancar el sistema se llama la 'partición raiz'/'root partition', la cual es (la mayoría de las veces) la única desde la cual usted puede iniciar el sistema. En ella se encuentra le directorio raíz del sistema de archivos, el nodo superior bajo el cual cuelga todo lo demás.

Las otras particiones en el sistema tienen que ser adjuntadas a la partición root para que la totalidad del sistema de múltiples particiones pueda ser accesible. Él *montará* cada una de las particiones en un directorio en la partición root.

Por ejemplo, si usted tiene un directorio UNIX llamado /usr, es probable que este constituya sea un punto de montaje que apunta a una partición que contiene muchos programas que han sido instalados con el UNIX pero que no son requeridos durante el proceso de arranque.

10.4. ¿Cómo un archivo logra ser visto?

Ahora podemos dar una mirada al sistema de archivos desde arriba. Cuando usted abre un archivo (por ejemplo: /home/fulano/www/ldp/fundamentals.xml) esto es lo que sucede.

El kernel arranca en la raíz el sistema de archivos de UNIX (en la partición raíz). Busca un directorio llamado 'home'. Usualmente 'home' es un punto de montaje para la partición de usuario que esta ubicada en cualquier otro lugar, así que el kernel se dirige hacia ese lugar. En el nivel superior de la estructura del directorio de dicha partición de usuario, el kernel busca una entada llamada 'fulano' y extrae un número i-node. El kernel entonces se dirige a dicho i-node, advirtiendo que dicho i-node se encuentra asociado a un bloque de datos de archivos correspondiente a una estructura de archivos. Aquí busca 'WWW', extrae *dicho* número i-node y se dirige al correspondiente subdirectorio, buscando ahora 'lpd'. Aquí hay otro i-node de directorio. Al abrirlo encuentra el numero i-node para 'fundamentals-xml'. Ese i-node no es un directorio, en lugar de ello dicho i-node contiene la lista de bloques del disco asociados con el archivo.

10.5. Propietario, permisos y seguridad de archivo

UNIX tiene características de permisos para mantener a los programas alejados de accidentes o malas situaciones al tomar datos que no debe usar. Estas características fueron originalmente diseñadas para soportar el timesharing (compartición del tiempo) al proteger en la misma máquina a múltiples usuarios de entre si, en la época en la que UNIX funcionaba principalmente en costosas minicomputadoras compartidas.

Para entender el concepto de archivos de permisos es necesario recordar la descripción de usuarios y grupos en la sección ¿Que pasa cuando usted se registra en el sistema?. Cada archivo tiene un usuario propietario y un grupo propietario. Inicialmente estos permisos son los del creador del archivo: ellos pueden ser cambiados con los programas chown(1) y chgrp(1).

Los permisos básicos que pueden ser asociados con un archivo son '**read**' (permiso para leer sus datos), '**write**' (permisos para modificarlo) y '**execute**' (permisos para correr como un programa). Cada archivo tiene tres conjuntos de permisos: uno para su usuario propietario, uno para su grupo propietario, y uno para los demás. Los 'privilegios' que usted logra cuando se registra en el sistema son sólo la capacidad de lectura, escritura y ejecución de aquellos archivos para los cuales los bits de permisos están asignados a su 'identificador de usuario'/'user ID' o a uno de los grupos a los que pertenece, o archivos que han sido adecuados como accesibles para todo el mundo.

Observemos una lista de archivos en un hipotético sistema UNIX para ver como estos permisos interactúan y como UNIX los despliega.

```
snark:~$ ls -l notes

-rw-r--r-- 1 fulano users 2993 Jun 17 11:00 notes
```

Este es un archivo ordinario de datos. La lista nos indica que pertenece al usuario 'fulano' y fue creado dentro de su grupo propietario 'users'. Probablemente la máquina en la que estamos ubico

por defecto en ese grupo a cada usuario ordinario: otros grupos de usuario que comúnmente usted podrá ver son 'staff', 'admin', 'wheel' (por obvias razones los grupos no son muy importantes en computadoras personales o estaciones de trabajo unipersonales). Su sistema UNIX puede usar un grupo por defecto distinto, quizás alguno que tenga su propio nombre de identificador de Usuario.

La cadena '-rw-r—r--' representa los bits de permisos para el archivo. El primer bit es la posición para el bit del directorio: El primer - es la posición para el bit de directorio, mostrará 'd' si el archivo es un directorio, o mostrar 'l' si el archivo es un link simbólico. A continuación de esto se presentan los siguientes tres grupos de permisos, y el tercer grupo son permisos para los demás (a menudo llamados 'permisos para todo el mundo'). En este archivo el usuario propietario 'fulano' puede leer o escribir el archivo, otras personas en el grupo 'users' pueden leer el archivo y cualquier otra persona en el mundo puede leerlo. Este es el conjunto de permisos típico para un archivo ordinario de datos.

Ahora vamos a mirar un archivo con unos permisos muy diferentes. Este archivo es el GCC (el compilador GNU del lenguaje de programación C)

```
snark:~$ ls -l /usr/bin/gcc

-rwxr-xr-x 3 root bin 64796 Mar 21 16:41 /usr/bin/gcc
```

Este archivo pertenece al usuario llamado 'root' y a un grupo llamado 'bin': este archivo puede ser escrito (modificado) solo por el usuario 'root' ,pero leído o ejecutado por cualquier otro. Esta es una típica titularidad y configuración de permisos para un comando pre-instalado en el sistema. El grupo 'bin' existe en algunos UNIX para agrupar juntos a los comandos del sistema (el nombre es una reliquia histórica, es la abreviatura para 'binario'). En vez de el grupo 'usr' el sistema UNIX podría usar el grupo 'root' (¡no es exactamente lo mismo que el usuario 'root'!).

El usuario 'root' es el nombre convencional para el usuario con el numero de identificación 0, que es una especial y privilegiada cuenta que puede sobrescribir todos los privilegios. El acceso de usaurio root es útil pero peligroso: un error al teclear mientras se esta como 'root' adentro del sistema puede alterar archivos críticos del sistema que un usuario ordinario no podría tocar usando el mismo comando.

Debido a que la cuenta root es muy poderosa, se hace necesario cuidar el acceso a ella y protegerlo cuidadosamente. La clave del usuario 'root' es el componente de seguridad más crítico de su sistema, y es a lo que muchos crackers e intrusos siempre desean tener acceso.

Sobre contraseñas: No la deje anotada ni escoja contraseñas que puedan ser adivinadas fácilmente (como el primer nombre de su novio/a esposo/a). Esto constituye una práctica muy mala pero común, que ayuda a los crackers en gran medida. En general, no escoja ninguna palabra del diccionario, puesto que hay programas llamados 'diccionarios de crackers' que funcionan buscando contraseñas probables gracias a listados de palabras comunes elegidas. Una buena técnica consiste en escoger una combinación consistente en una palabra, un número, y otra palabra, como por ejemplo 'tiburon6sidra' o 'yerba7serpiente': esto hará que el tiempo de búsqueda requerido para un diccionario sea muy largo. No use estos necesariamente estos ejemplos, ya que es esperable que los crackers después de leer este documento incorporen estos ejemplos en sus diccionarios.

Ahora veamos un tercer caso:

```
snark:~$ ls -ld ~
```

```
drwxr-xr-x 89 fulano users 9216 Jun 27 11:29 /home2/fulano

snark:~$
```

Este archivo es un directorio (note el 'd' en el primer espacio de permisos). Observamos que el archivo puede ser escrito por fulano pero leído y ejecutado por cualquier otro.

El permiso de lectura sobre un directorio le da la habilidad de listar el directorio – esto consiste en poder ver los nombres de los archivos y directorios que él contiene. El permiso de escritura da la habilidad de crear y borrar archivos en el directorio. Si usted recuerda que el directorio incluye una lista de nombres de los archivos y subdirectorios que contiene, estas reglas cobrarán sentido

Los permisos de ejecución en un directorio significa que es posible entrar en el directorio para abrir los archivos y los subdirectorios contenidos en él. En efecto, le da permisos para acceder a los i-nodes en el directorio. Un directorio sin ningún permiso de ejecución ser{ia inútil.

Ocasionalmente usted podría ver un directorio que es ejecutable para los usuarios 'world' pero no es legible para estos mismos usuarios: Esto significa que un usuario aleatorio podrá acceder a los archivos y directorios inferiores pero sólo si conoce sus nombres exactos (ya que el directorio no puede ser listado).

Es importante recordar que los permisos *read*, *write*, o *execute* de un directorio es independiente de los permisos en los archivos y directorios que éste contiene. En particular, la posibilidad de escritura en un directorio no da automáticamente acceso de escritura a los archivos existentes.

Finalmente vamos a ver los permisos del programa login.

```
snark:~$ ls -l /bin/login

-rwsr-xr-x 1 root bin 20164 Apr 17 12:57 /bin/login
```

Este programa tiene los permisos que podríamos esperar para un comando de sistema – excepto por la 's' donde el permiso de ejecución para el propietario debería estar. Esta es la manifestación visible de un permiso especial llamado el 'set-user-id' o 'setuid bit'

El bit setuid es normalmente adjuntado a programas que necesitan dar privilegios de root a usuarios ordinarios, pero de una forma controlada. Cuando este permiso es definido en un programa ejecutable usted logra los privilegios del propietario del archivo del programa mientras el programa corra por parte suya, ya sea o no que esto coincida con sus privilegios.

Como la cuenta root misma, los programas setuid son útiles pero peligrosos. Quienquiera que pueda subvertir o modificar un programa setuid propiedad del root puede utilizarlo para generar una shell con privilegios de usuario. Por esta razón abrir un archivo en la mayoría de los UNIX para escribirlo, provoca que que dicho archivo desactive su bit setuid. Muchos ataques a la seguridad de UNIX tratan de explotar bugs en los programas setuid buscando aprovecharse de ellos. Por lo tanto, los administradores de seguridad de sistemas deben prestar constante atención a dichos programas, siendo cuidadosos y reacios a instalar nuevos.

Debemos resaltar un par de importantes detalles de lo que hemos visto ahora: A saber, como el grupo propietario y sus permisos son asignados cuando un archivo o directorio es creado por primera vez. El tema de grupos es de cuidado porque los usuarios pueden ser miembros de múltiples grupos, pero uno de ellos (especificado en la entrada del usuario en /etc/passwd) es el grupo por defecto para el usuario y normalmente será el propietario de los archivos creados por el usuario.

La historia detrás de los bits de permisos con los que inicia un archivo es un poco mas complicada. Un programa que crea un archivo normalmente especificará los permisos con los que dicho archivo comenzará. Pero estos permisos serán modificados por una variable de entorno del usuario denominada 'umask'. El umask especifica cuales bit de permisos *serán desactivados* al crearse un archivo; el valor más común - y que esta establecido por defecto en la mayoría de sistemas- es -------w- o 002, el cual desactiva el bit de la escritura para todo el mundo. Para mayores detalles mire la documentación del comando umask en su shell.

El grupo inicial para un directorio es también un poco mas complicado. En algunos sistemas UNIX un nuevo directorio obtiene el grupo por defecto del usuario creador (esta es una convención del UNIX System V): en otros, se obtiene el grupo propietario del directorio padre en el cual ha sido creado (esta es la convención BSD). En algunos UNIX modernos, incluyendo Linux, la conducta definitiva puede ser seleccionada al configurar el set-group-ID en el directorio (chmod g+s).

10.6. ¿Cómo las cosas pueden andar mal

Anteriormente se aludió que los sistemas de archivos eran frágiles. Ahora sabemos que para obtener un archivo usted tiene que saltar a través de lo que podría ser una arbitraria y larga cadena de directorios y referencias de i-node. Ahora supongamos que su disco rígido empeora con un sector dañado.

Si usted tiene suerte, solo desechará algunos datos de archivo. Si no tiene suerte, se podría corromper una estructura de directorio o un numero i-node y dejar en el limbo a la totalidad del árbol de un subdirectorio, o peor aún, resultar en una estructura corrompida que apunta múltiples rutas a un mismo bloque de disco o i-node. Tal daño puede propagarse por una operación de archivo normal, desechando datos que no estaban en el lugar donde inicio el sector dañado.

Afortunadamente, este tipo de contingencias se han convertido en algo poco común así como el hardware de los discos se han vuelto más confiables. Aun más, esto significa que UNIX buscará revisar periódicamente la integridad del sistema de archivos para asegurarse que todo funcione correctamente. Los UNIX modernos realizan rápidamente dicha revisión de integridad en cada partición al momento del arranque, justo antes de montarlas. Cada pocos re inicios sucederá la revisión de integridad de manera mas detallada, tomándose unos pocos minutos de duración.

Si todo esto suena a que UNIX es terriblemente complejo y susceptible de fallas, entonces puede sonar tranquilizador saber que estas revisiones al momento del arranque comúnmente detectan y corrigen problemas normales *antes* de que ellos se conviertan en problemas realmente desastrosos. Otros sistemas operativos no tienen estas facilidades, lo cual aligera un poco el tiempo de arranque pero puede dejarlo a usted mas seriamente afectado cuando intente recuperarlo a mano (inclusive asumiendo que usted tiene a mano una copia de las Utilidades Norton o cualquier otra cosa).

Una de las tendencias en los diseños de los actuales UNIX es el 'registro diario del sistema de archivos'/'journalling file systems'. Este sistema organiza el tráfico para el disco garantizando que el

disco se encuentre en un consistente estado al cual pueda regresar. Esto hace que la revisión de integridad detallada sea mas rápida en el momento del arranque.

inicio

11. ¿Como trabajan los lenguajes de programación?

Ya hemos discutido sobre como se ejecutan los programas. En última instancia cada programa deba ser ejecutado como un flujo de datos formado por instrucciones en el lenguaje máquina de su computadora. Pero los seres humanos no manejan muy bien el lenguaje de máquina; hacerlo se ha convertido en un oscuro arte legendario aún entre los hackers.

Hoy en día casi todo el código de UNIX está escrito en un 'lenguaje de alto nivel'/'high-level language' excepto una pequeña cantidad de interfaces directas de hardware soportadas en el kernel mismo. (El término 'alto nivel' es un viejo término usado para distinguir los 'lenguajes ensamblador'/'assembler languajes' de 'bajo nivel'/"low-level, los cuales son básicamente pequeños envoltorios alrededor del código máquina.

Existen varios tipos distintos de lenguajes de alto nivel. Para hablar acerca de ellos es útil tener en mente que el 'código fuente'/'source code' de un programa (la versión editable creada por un ser humano) tiene que pasar a través de algún tipo de traducción al código máquina para que la máquina pueda realmente correrlo.

11.1. Lenguajes Compilados

La mayoría de los tipos de lenguaje convencionales son lenguajes compilados. Estos son traducidos a archivos ejecutables de código máquina binario por un programa especial denominado - de forma lógica - compilador. Una vez ha sido generado el binario, usted puede hacerlo correr directamente sin tener que mirar nuevamente el código fuente (la mayoría del software resulta distribuido para su entrega en binarios ya compilados, hechos a partir de código que usted no ve).

Los lenguajes compilados tienden a dar un excelente desempeño y a contar con un acceso mas completo al sistema Operativo, pero también hay dificultad en programar en él.

C - el lenguaje en el que está escrito UNIX - es por lejos es el mas importante de ellos (con su variante **C++**). **FORTRAN** es otro lenguaje compilado que aún es usado entre ingenieros y científicos pero con una edad que lo supera en años, y mucho mas primitivo. En el mundo UNIX estos son los lenguajes compilados que principalmente se usan: Fuera de ellos, **COBOL** es ampliamente usado para software de finanzas y negocios.

Solían existir muchos otros lenguajes compilados, pero la mayoría de ellos se ha extinguido o son estrictamente usados como herramientas de investigación. Si usted es un nuevo desarrollador de UNIX usando un lenguaje compilado, estará probablemente usando C o C++.

11.2. Lenguajes Interpretados

Un lenguaje interpretado depende de un programa intérprete que lea el código fuente y lo traduzca inmediatamente en cómputos y llamadas del sistema. El código fuente debe ser reinterpretado (y el intérprete estar presente) toda vez que el código requiera ser ejecutado.

Los lenguajes interpretados tienden a ser más lentos que los lenguajes compilados, y a menudo tienen acceso acceso al hardware subyaciente y a partes importantes del sistema operativo. Por otro lado, estos lenguajes tienden a ser más fáciles para programar y menos estrictos que los lenguajes compilados

Muchas utilidades de UNIX, incluyendo su shell, bc, sed y awk constituyen - de hecho - pequeños intérpretes de lenguajes. **BASIC** es usualmente interpretado, de igual manera **Tcl**. Históricamente, el más importante lenguaje interpretado ha sido **LISP** (una gran mejora sobre la mayoría de sus sucesores). Hoy, las shells de UNIX y el Lisp que se encuentra dentro del editor Emacs son probablemente los mas importantes lenguajes interpretados puros.

11.3. Lenguajes Pseudocódigo

Desde 1990 han ido cobrando mayor importancia un tipo de lenguaje híbrido que usa la compilación y la interpretación. los lenguajes *P-code* guardan similitud a los lenguajes compilados en el hecho que el código fuente resulta traducido a un binario compacto que será lo que ejecute realmente, pero no es un código máquina. En lugar de éste, te utiliza un pseudocódigo (o *código-p*) que normlamente es bastante mas simple pero más poderoso que un código máquina real. Cuando usted corre el programa, se interpreta el p-código.

El pseudocódigo puede correr casi tan rápido como un binario compilado (los interpretes de pseudocódigo pueden ser simples, sencillos y rápidos). Pero los lenguajes pseudocódigo pueden mantener la flexibilidad y poder de un lenguaje interpretado.

Algunos lenguajes pseudocódigo importantes son **Python**, **Perl** y **Java**.

inicio

12. Que hace que Internet trabaje?

Para ayudar a entender cómo funciona Internet vamos a mirar las cosas que suceden cuando usted hace una tarea normal en Internet – apuntar con el navegador hasta la página de inicio de este documento en el sitio web del Proyecto de documentación de Linux (Linux Documentation Project - LDP). Este documento es:

https://tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/internet.html

Lo que significa que el se encuentra en el archivo HOWTO/UNIX-and-Internet-Fundamentals-HOWTO/index.html bajo el directorio de exportación World Wide Web accesible en la telaraña de alcance global en el host www.tldp.org

12.1. Nombres y Ubicaciones

La primer cosa que su navegador tiene que hacer es establecer una conexión con la máquina donde se encuentra el documento. Para hacer esto, primero debe buscar en la red la ubicación de el *host* www.tldp.org ('host' se usa para abreviar 'máquina anfitriona'/'host machine' o 'red anfitriona'/'network host': www.tlpd.org es un *hostname* típico). Su ubicación corresponde realmente a un número denominado 'Dirección IP'/'IP adress' (explicaremos que significa IP más adelante).

Para hacer esto el navegador interroga a un programa llamado 'Servidor de nombres'/'name server'. El servidor de nombres puede encontrarse en su máquina, pero es más probable que funcione en un servidor con el cual su navegador se comunica. Cuando usted ingresa con un ISP (Proveedor de servicios de internet / Internet Service Provider . ISP. n del t.), parte del procedimiento de configuración la mayoría de las veces involucrará indicarle al navegador la dirección IP del servidor de nombres en la red.

Los servidores de nombres ubicados en diferentes máquinas hablan entre si, intercambiando información y manteniendo actualizada toda la información que necesitan para resolver los nombres de host (relacionarlos con las direcciones IP). El servidor de nombres que usted usa, en el proceso de resolver la ubicación de www.tldp.org, podría tener que consultar a tres o cuatro sitios diferentes a través de la red, pero esto usualmente sucede muy rápido (en menos de un segundo). Veremos en detalle a los servidores de nombres en la próxima sección.

El servidor de nombres le dirá a su navegador que la dirección IP del host www.tldp.org es 152.19.254.81. Conociendo esta información crucial, el navegador de nuestra máquina será capaz de intercambiar bits directamente con www.tldp.org.

12.2. El Sistema de Nombres de Dominio

La totalidad de los programas y bases de datos de red que cooperan en resolver los nombres de host a direcciones IP son llamados 'DNS' (Sistema de Nombres de Dominio). Cuando usted ve referencias a un 'servidor DNS'/'DNS Server' esto significa que estamos hablando de un servidor de dominios. Ahora vamos a ver como funciona todo esto.

Los nombres de máquinas en la Internet están compuestos de partes separadas por puntos. Un 'dominio'/'domain' es una colección de máquinas que comparten un sufijo como nombre en común. Estos Dominios pueden residir dentro de otros dominios. Por ejemplo, la máquina www.tldp.org se encuentra en el subdominio .tldp.org del dominio .org.

Cada dominio esta definido por un servidor de nombres autorizado que conoce la dirección IP de las otras máquinas en el dominio. El servidor de nombres autorizado (o primario) puede tener backups en caso de alguna falla: si usted ve referencias a un 'servidor de nombres secundario'/'sencondary name server' (o 'secundario DNS'/'secondary DNS') es que se esta hablando de uno de esos bakups. Estos servidores secundarios a menudo refrescan su información desde sus servidores primarios cada pocas horas, así, un cambio hecho en el servidor primario a la relación nombre de host-IP resultará propagado automáticamente.

Ahora, aquí esta la parte importante. Los servidores de nombres de un dominio *no tienen que conocer* la ubicación de todas las máquinas en otros dominios (incluyendo sus propios subdominios): sólo es necesario qeu conozcan la ubicación de los servidores de nombres. En

nuestro ejemplo, el servidor de nombres autorizado para administrar el dominio .org sabe la dirección IP de el servidor de nombres para .tlpd.org pero no la dirección de todas las demás máquinas en el dominio .tlpd.org.

Los dominios en el sistema DNS están organizados como un gran árbol invertido. En la cima se encuentran los 'servidores raíz'/'root servers' (administran los dominios .org .com. net, etc. n del t): ellos están conectados a su software DNS. El servidor raíz sabe las direcciones IP de los servidores de dominio para los 'dominios de nivel superior'/'top-levels domains' (tlpd.org, ejemplo.org, etc. n. del t.). Cada servidor de dominios de nivel superior conoce donde se encuentran los servidores de dominios directamente por debajo ellos, y así sucesivamente.

Los DNS están cuidadosamente diseñados para que cada máquina pueda funcionar con independencia con la mínima cantidad de conocimientos que necesite sobre la forma total del árbol, y para que los cambios locales en las ramas inferiores puedan realizarse de forma simple alterando en la base de datos de un servidor de nombres autorizado los registros 'nombre de host'-'dirección IP'.

Cuando su navegador hace la consulta sobre la dirección IP de www.tlpd.org lo que sucede realmente es: primero su servidor de nombres (quizás el servidor del ISP de su cuenta de acceso a Internet. n. del t.) consulta al servidor raíz para saber dónde puede ubicar los servidores de nombres para los dominios .org. Una vez conoce esto, él entonces pregunta al servidor de nombres del dominio .org para saber donde la dirección IP de el servidor de nombres de .tldp.org. Una vez se ha hecho esto él pregunta al servidor de nombres .tldp.org para saber la dirección del host www.tldp.org.

La mayoría de las veces, su servidor de nombres ni siquiera tiene que trabajar tanto. Los servidores de nombres realizan mucho caché: cuando su servidor resuelve un nombre de host, mantiene en memoria por un momento dicha asociación con la dirección IP resultante. Esto es el motivo por el cual, cuando usted navega a un nuevo sitio web usualmente usted ve un mensaje de su navegador que indica 'buscando'/'looking up' del host para la primera página que solicitó. Eventualmente el proceso de mapeo Hostname/Dirección IP expira, y su DNS tendrá que solicitar nuevamente este proceso – esto es importante para que no se conserve la informaciòn antigua (e inválida) cuando un nombre de host cambie de dirección. La dirección IP cacheada de un sitio web no será desplegada si el host es inalcanzable.

12.3. Paquetes y enrutadores

Lo que el navegador querrá hacer es enviar un comando al servidor web en www.tldp.org que se parece a esto.

```
GET /LDP/HOWTO/Fundamentals.html HTTP/1.0

```

Esto es lo que sucede: el comando es convertido en un 'paquete'/'packet', un bloque de bits como un telegrama que es empaquetado junto a tres cosas importantes: la *dirección origen* (la dirección de su máquina), la *dirección destino* (152.19.254.81), y un *número de servicio* o *número de puerto* (80 en este caso) que indica que es una solicitud hecha para la telaraña mundial World Wide Web.

Su máquina entonces envíará el paquete por el cable (ya sea a la conexión de su ISP o red local) hasta que llega a una máquina especializada llamada 'enrutador'/'router'. El enrutador tiene un mapa

de Internet en su memoria – no siempre completo, pero describe por completo el vecindario de su red y sabe cómo conseguir los enrutadores de otros vecindarios en la Internet.

Su paquete puede pasar a través de varios enrutadores al encaminarse a la dirección de destino. Los enrutadores son inteligentes. Ellos observan cuanto tardan otros enrutadores en avisar que han recibido un paquete. También usan esa información para dirigir el tráfico sobre conexiones rápidas. Los enrutadores usan estas conexiones para advertir cuando otro enrutador(o un cable) se ha desconectado de la red y compensarlo - si es posible - dando con otro enrutador.

Existe una leyenda urbana que dice que la Internet fue diseñada para sobrevivir a una guerra nuclear. Esto no es cierto, pero el diseño de la Internet es extremadamente bueno en conseguir un desempeño confiable sobre hardware problemático en condiciones inciertas. Esto se debe principalmente al hecho de que su inteligencia está distribuida a través de miles de enrutadores en lugar de estar concentrada en unos pocos conmutadores centrales gigantes (como los de la red telefónica). Esto significa que las fallas tienden a ser bien localizadas y existen circuitos alternativos que pasan alrededor de estas fallas.

Una vez que su paquete logra llegar a la máquina destino, esta máquina usa el numero de servicio del mismo para alimentar el paquete con la solicitud al servidor web. El servidor web conocer a dónde enviar respuesta revisando la dirección IP origen en el paquete del comando. Cuando el servidor web devuelve este documento (el que estas leyendo. n del t.), lo hará dividiéndolo en paquetes numerados. El tamaño de estos paquetes variará según el medio de transmisión en la red y el tipo de servicio.

12.4. TCP e IP

Para entender como se lleva a cabo una transmisión de paquetes mñultriples, usted necesita saber que la Internet actualmente usa dos protocolos (maneras de negociar e intercambiar la información n.del t.), organizados uno sobre otro.

En el nivel inferior, el IP ('Protocolo de Internet' /'Internet Protocol'), es el responsable de etiquetar individualmente los paquetes con las direcciones fuente y destino de dos computadoras que intercambian información sobre una red. Por ejemplo, cuando usted accede a http://www.tldp.org, el paquete que usted envió tiene la dirección IP de su computadora (una como 192.168.0.101) y la dirección 152.19.254.81 de la computadora www.tldp.org. Estas direcciones funcionan de la misma manera que la dirección de su hogar cuando alguien envía a usted correspondencia. La oficina postal puede leer la dirección y determinar donde se encuentra usted y la mejor ruta para hacerle llegar su carta, de manera similar a la que un enrutador lo hace con el trafico de Internet.

En el nivel superior, TCP (Protocolo de Control de Transmisión'/'Transmission Control Protocol') le da confiabilidad. Cuando dos máquinas negocian una conexión TCP (lo que hacen empleando el protocolo IP), el receptor sabe enviar acuses de recibo del paquete que recibió del remitente. Si el remitente no recibe un acuse de recibo del paquete en un tiempo de espera determinado, reenviará otra copia del paquete. Adicionalmente, el remitente otorga a cada paquete TCP una secuencia de números, que el receptor puede usar durante el rearmado de los paquetes en caso de que ellos lleguen en desorden (es probable que esto suceda si la velocidad de red varía durante la conexión).

Los paquetes IP también contienen un checksum (comprobación o suma de comprobación. n. del t.), procedimiento para facilitar la detección de datos corruptos debido a problemas en la red. (El checksum se computa a partir del resto del paquete de tal forma que si el resto del paquete o el

checksum están corruptos, al realizar nuevamente dicho cálculo y comparación sea extremadamente probable reconocer un error). Así, desde el punto de vista de cualquiera que usa TPC/IP y los servidores de nombre, esta parece una forma confiable de enviar flujos de bytes entre pares de nombre-de-host/numero-de-servicio. Las personas que escriben protocolos de red casi nunca tienen que pensar sobre la definición del tamaño del paquete, como se rearma el paquete, la revisión de errores, el procedimiento checksum y la retransmisión que sucede en un nivel inferior.

12.5. HTTP, un protocolo de aplicación

Ahora vamos a regresar a nuestro ejemplo. Los navegadores y servidores web hablan un 'protocolo de aplicación'/'application protocol' que funciona encima de TCP/IP, usándolo simplemente como una forma de enviar y recibir flujos de bytes. Este protocolo se llama *HTTP ('Protocolo de Transferencia de Hipertexto'/'Hypertext Transfer Protocol'*) y ya hemos visto un comando de él – el GET mostrado anteriormente.

Cuando el comando GET interactúa con el servidor web de www.tldp.org con el puerto de servicio número 80, será despachado a un 'servidor demonio'/'server daemon' que escucha en el puerto 80. La mayoría de los servicios de Internet son implementados por servidores demonio que no hacen más que escuchar en los puertos, esperando y ejecutando órdenes que arriben allí.

Si el diseño de la Internet tiene una regla que todo lo rige, esta es que todas las partes deben ser tan simples y accesibles a seres humanos como sea posible. HTTP y sus parientes (como el Simple Mail Transfer Protocol SMTP usado para mover electrónicamente correo entre hosts) tienden a recurrir a comandos de texto simples de imprimir que terminan con un retorno de carro/alimentación de línea.

Esto es ligeramente ineficiente: en algunas circunstancias podría obtenerse mayor velocidad empleando un protocolo de código estrictamente binario. Pero la experiencia ha demostrado que los beneficios de tener comandos fáciles de describir y entender por seres humanos sobrepasa cualquier beneficio en eficiencia que podrían obtenerse al costo de de hacer las cosas difíciles y opacas.

De la misma forma, lo que el servidor demonio que le devuelve lo solicitado por usted vía TCP/IP también operae texto. El comienzo de la respuesta puede parecerse a algo como esto:

```
HTTP/1.1 200 OK

Date: Sat, 10 Oct 1998 18:43:35 GMT

Server: Apache/1.2.6 Red Hat

Last-Modified: Thu, 27 Aug 1998 17:55:15 GMT

Content-Length: 2982

Content-Type: text/html
```

A este encabezado seguirá a continuación una línea en blanco y el texto de la página web (después de lo cual la conexión será interrumpida). Su navegador solo presentará dicha página. Las cabeceras le indicarán al navegador cómo hacerlo (en particular, la cabecera Content-Type indica que el dato de retorno realmente es un documento hipertextual html.

inicio

13. Para aprender más

Esta es una lista de lecturas de HOWTO que contiene libros que usted puede leer para aprender más sobre las temáticas que hemos aquí abarcado. Quizás también quiera leer el documento Cómo convertirse en un Hacker.

Historia de Revisiones
Revision 2.14
Actualizaciones menores.
Revision 2.13
Actualizaciones menores.
Revisión 1.0
Revisión Inicial.